МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «Санкт-Петербургский национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

**ЛАБОРАТОРНАЯ РАБОТА №4**

по дисциплине «Информационные системы и базы данных»

Вариант № 1675

***Выполнил:***

Студент группы P3116

Билошицкий Михаил Владимирович

***Преподаватель:***

Горбунов Михаил Витальевич

Санкт-Петербург, 2023

# Содержание

[Содержание 2](#_Toc137301441)

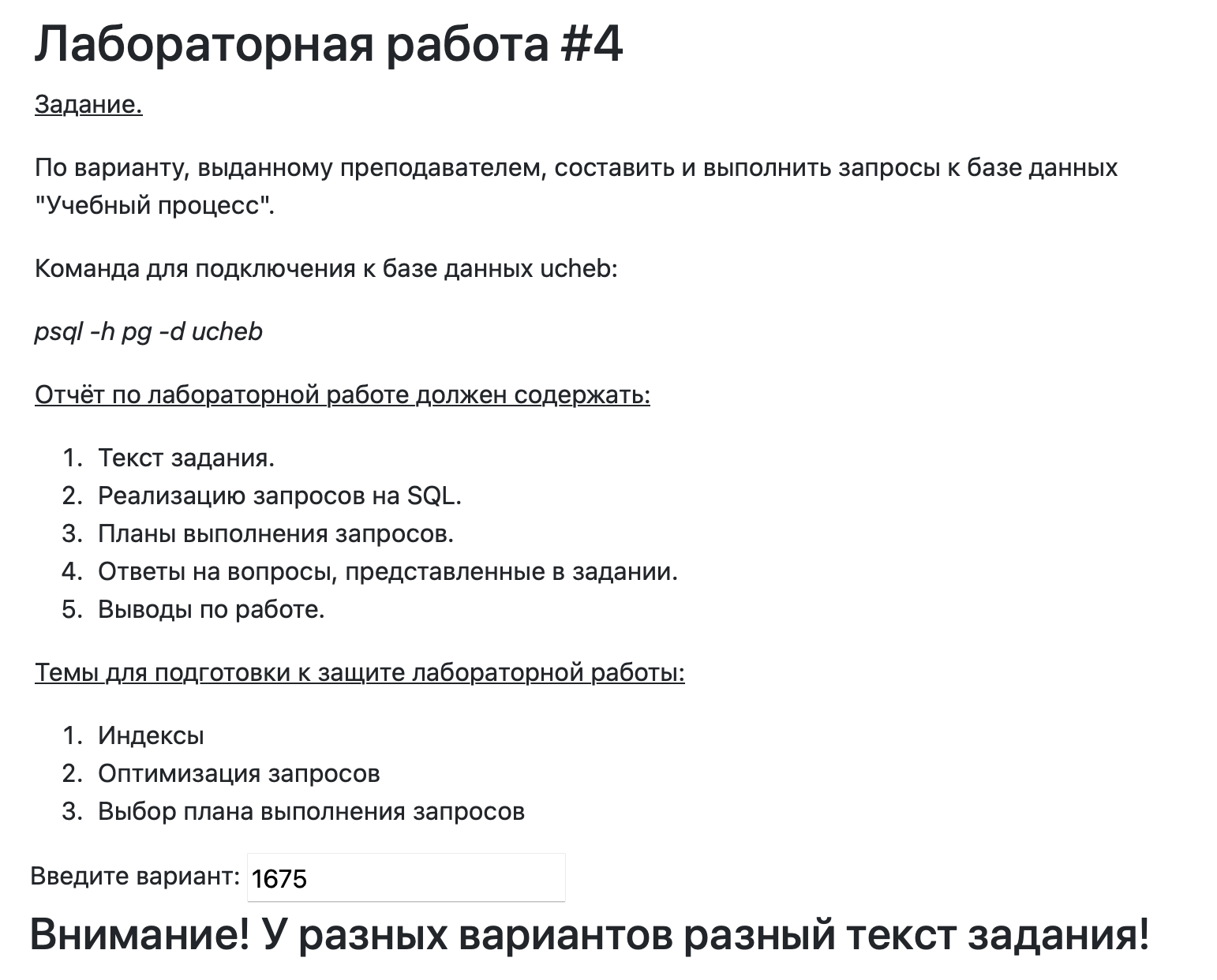
[Текст задания 3](#_Toc137301442)

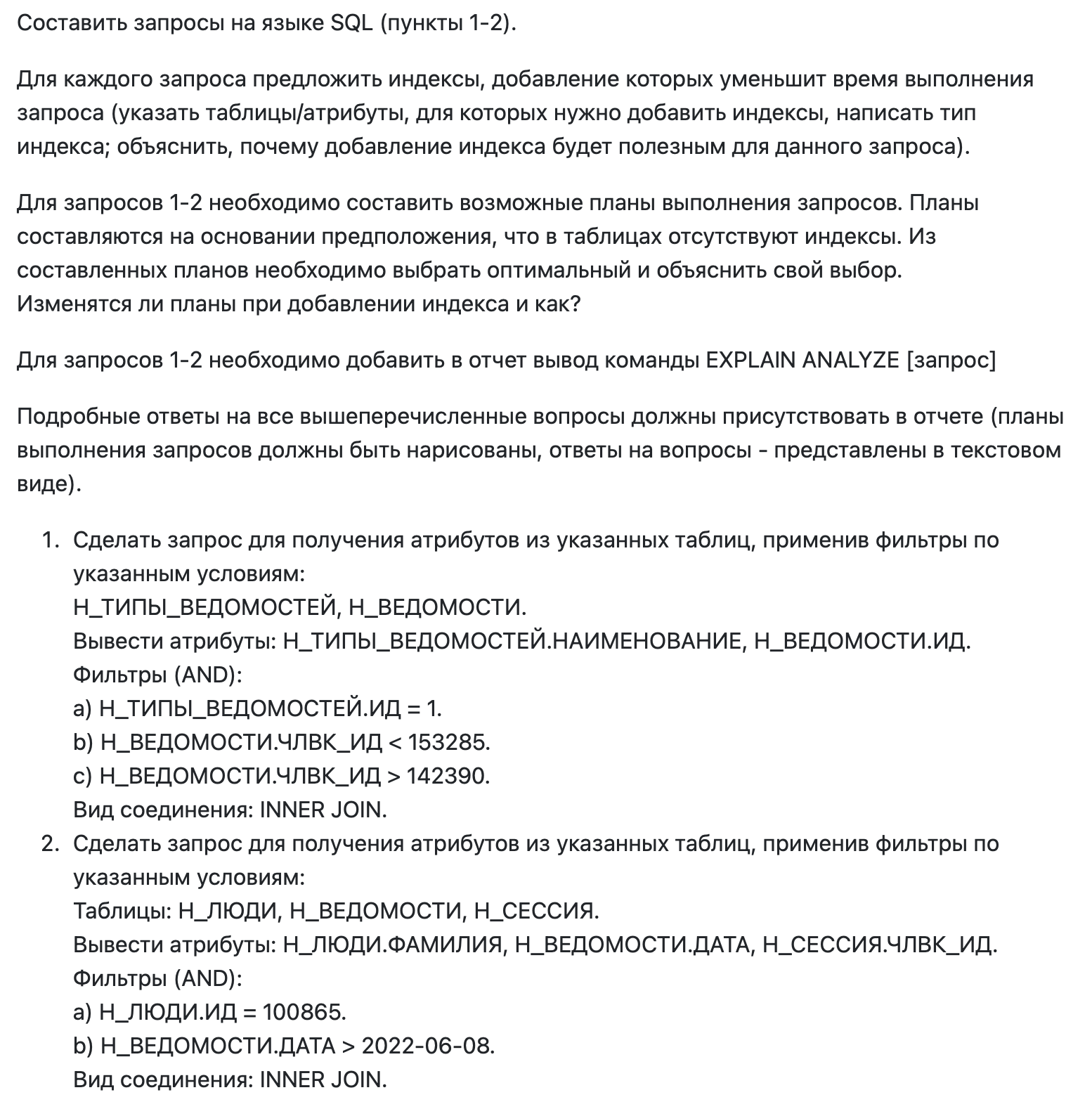
[Реализация запросов на SQL 4](#_Toc137301443)

[Планы выполнения запросов 6](#_Toc137301444)

[Вывод 8](#_Toc137301445)

# Текст задания





# Реализация запросов на SQL

**Запрос 1.**

Индексы:

* Таблица Н\_ТИПЫ\_ВЕДОМОСТЕЙ атрибут ИД, индекс **не будет эффективен**, так как таблица имеет небольшое количество строк и добавление индекса только замедлит выполнение запроса.
* Таблица Н\_ВЕДОМОСТИ атрибут ТВ\_ИД, индекс **не будет эффективен**, так как атрибут ТВ\_ИД является внешним ключом таблицы Н\_ТИПЫ\_ВЕДОМОСТЕЙ атрибута ИД, которое содержит малое количество строк, следовательно, атрибут ТВ\_ИД содержит малое количество уникальных значений и использование индекса не даст значительного ускорения запроса.
* Таблица Н\_ВЕДОМОСТИ атрибут ЧЛВК\_ИД, индекс b-tree. **Индекс полезен**, так как в таблице содержится большое количество значений и атрибут ЧЛВК\_ИД содержит большое количество уникальных значений, по которым будет производится дальнейшая фильтрация и индекс ускорит выполнение запроса. Используется индекс типа b-tree, так как данный тип индекса лучше осуществляет сортировку данных, чем индекс типа hash, что в нашем запросе полезно, так как происходит фильтрация числового атрибута с операторами больше-меньше.

~~CREATE INDEX~~ **~~idx\_типы\_ведомостей~~** ~~ON Н\_ТИПЫ\_ВЕДОМОСТЕЙ (ИД);~~

~~CREATE INDEX~~ **~~idx\_ведомости\_тв\_ид~~** ~~ON Н\_ВЕДОМОСТИ (ТВ\_ИД);~~

CREATE INDEX **idx\_ведомости\_члвк\_ид** ON Н\_ВЕДОМОСТИ (ЧЛВК\_ИД);

SELECT Н\_ТИПЫ\_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ, Н\_ВЕДОМОСТИ.ИД

FROM Н\_ВЕДОМОСТИ

JOIN Н\_ТИПЫ\_ВЕДОМОСТЕЙ ON Н\_ТИПЫ\_ВЕДОМОСТЕЙ.ИД = Н\_ВЕДОМОСТИ.ТВ\_ИД

WHERE

Н\_ТИПЫ\_ВЕДОМОСТЕЙ.ИД = 1 AND

Н\_ВЕДОМОСТИ.ЧЛВК\_ИД < 153285 AND

Н\_ВЕДОМОСТИ.ЧЛВК\_ИД > 142390;

Вывод EXPLAIN ANALYZE:

Nested Loop (cost=496.17..5502.69 rows=30924 width=422) (actual time=1.916..15.219 rows=28671 loops=1)

-> Seq Scan on "Н\_ТИПЫ\_ВЕДОМОСТЕЙ" (cost=0.00..1.04 rows=1 width=422) (actual time=0.019..0.022 rows=1 loops=1)

Filter: ("ИД" = 1)

Rows Removed by Filter: 2

-> Bitmap Heap Scan on "Н\_ВЕДОМОСТИ" (cost=496.17..5192.41 rows=30924 width=8) (actual time=1.893..11.767 rows=28671 loops=1)

Recheck Cond: (("ЧЛВК\_ИД" < 153285) AND ("ЧЛВК\_ИД" > 142390))

Filter: ("ТВ\_ИД" = 1)

Rows Removed by Filter: 7602

Heap Blocks: exact=1645

-> Bitmap Index Scan on "ВЕД\_ЧЛВК\_FK\_IFK" (cost=0.00..488.44 rows=36014 width=0) (actual time=1.656..1.657 rows=36273 loops=1)

Index Cond: (("ЧЛВК\_ИД" < 153285) AND ("ЧЛВК\_ИД" > 142390))

Planning Time: 0.835 ms

Execution Time: 16.650 ms

**Запрос 2.**

Индексы:

* Таблица Н\_ЛЮДИ атрибут ИД, индекс hash. **Индекс полезен**, так как используется соединение таблиц Н\_ЛЮДИ и Н\_ВЕДОМОСТИ по данному атрибуту, а также фильтрация через оператор =, соответственно будет осуществляться поиск уникального значения атрибута по большой таблице Н\_ЛЮДИ и при присоединении с Н\_ВЕДОМОСТИ, с чем быстро может справиться hash индекс, который работает эффективно при поиске данных.
* Таблица Н\_ВЕДОМОСТИ атрибут ЧЛВК\_ИД, индекс hash. **Индекс полезен**, так как используется соединение таблиц Н\_ЛЮДИ и Н\_ВЕДОМОСТИ по данному атрибуту.
* Таблица Н\_ВЕДОМОСТИ атрибут ДАТА, индекс b-tree. **Индекс полезен** так как в таблице содержится большое количество значений и атрибут ДАТА содержит большое количество уникальных значений, по которым будет производится дальнейшая фильтрация и индекс ускорит выполнение запроса. Используется индекс типа b-tree, так как данный тип индекса лучше осуществляет сортировку данных, чем индекс типа hash, что в нашем запросе полезно, так как происходит фильтрация с операторами больше-меньше.

CREATE INDEX **idx\_люди\_ид** ON Н\_ЛЮДИ USING HASH (ИД);

CREATE INDEX **idx\_ведомости\_члвк\_ид** ON Н\_ВЕДОМОСТИ USING HASH (ЧЛВК\_ИД);

CREATE INDEX **idx\_ведомости\_дата** ON Н\_ВЕДОМОСТИ (ДАТА);

SELECT Н\_ЛЮДИ.ФАМИЛИЯ, Н\_ВЕДОМОСТИ.ДАТА, Н\_СЕССИЯ.ЧЛВК\_ИД

FROM Н\_ЛЮДИ

JOIN Н\_ВЕДОМОСТИ ON Н\_ЛЮДИ.ИД = Н\_ВЕДОМОСТИ.ЧЛВК\_ИД

JOIN Н\_СЕССИЯ ON Н\_ЛЮДИ.ИД = Н\_СЕССИЯ.ЧЛВК\_ИД

WHERE

Н\_ЛЮДИ.ИД = 100865 AND

Н\_ВЕДОМОСТИ.ДАТА > '2022-06-08';

Вывод EXPLAIN ANALYZE:

Nested Loop (cost=0.86..20.21 rows=15 width=28) (actual time=0.036..0.037 rows=0 loops=1)

-> Nested Loop (cost=0.58..15.51 rows=1 width=28) (actual time=0.035..0.036 rows=0 loops=1)

-> Index Scan using "ЧЛВК\_PK" on "Н\_ЛЮДИ" (cost=0.28..8.30 rows=1 width=20) (actual time=0.032..0.032 rows=1 loops=1)

Index Cond: ("ИД" = 100865)

-> Index Scan using "ВЕД\_ДАТА\_I" on "Н\_ВЕДОМОСТИ" (cost=0.29..7.20 rows=1 width=12) (actual time=0.002..0.002 rows=0 loops=1)

Index Cond: ("ДАТА" > '2022-06-08 00:00:00'::timestamp without time zone)

Filter: ("ЧЛВК\_ИД" = 100865)

-> Index Only Scan using "SYS\_C003500\_IFK" on "Н\_СЕССИЯ" (cost=0.28..4.54 rows=15 width=4) (never executed)

Index Cond: ("ЧЛВК\_ИД" = 100865)

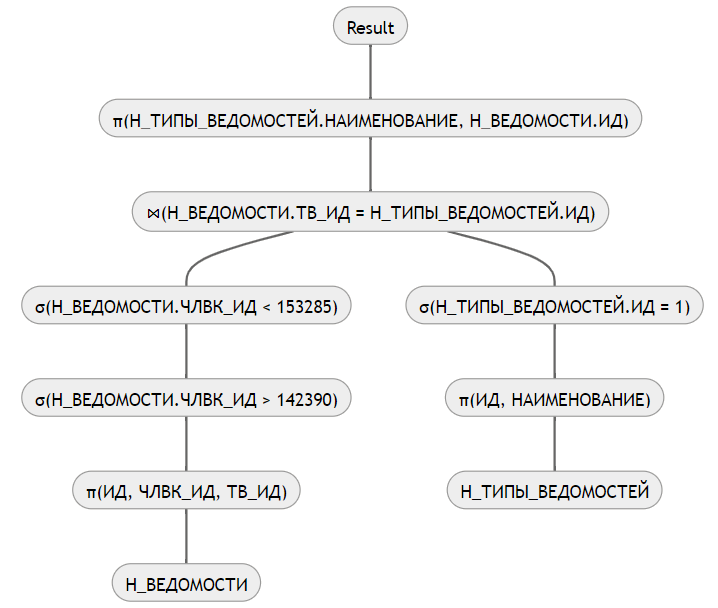
Heap Fetches: 0

Planning Time: 0.704 ms

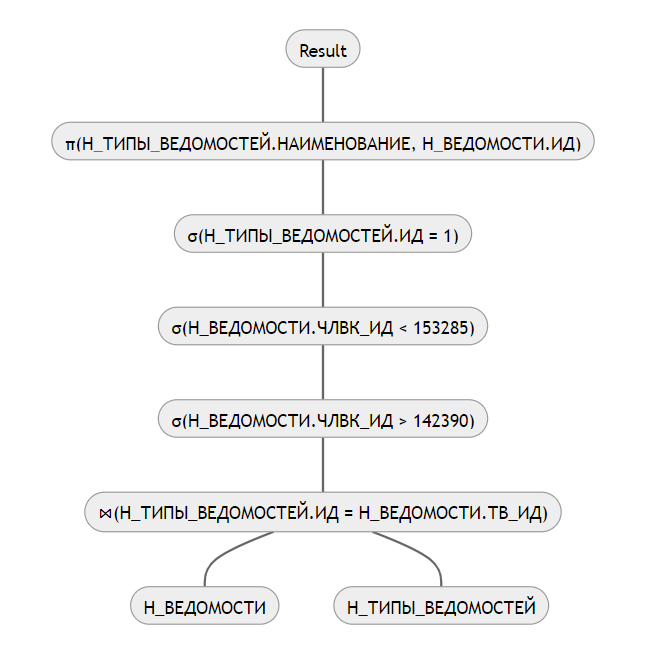
# Планы выполнения запросов

**Запрос 1.**

План 1.

****

План 2.

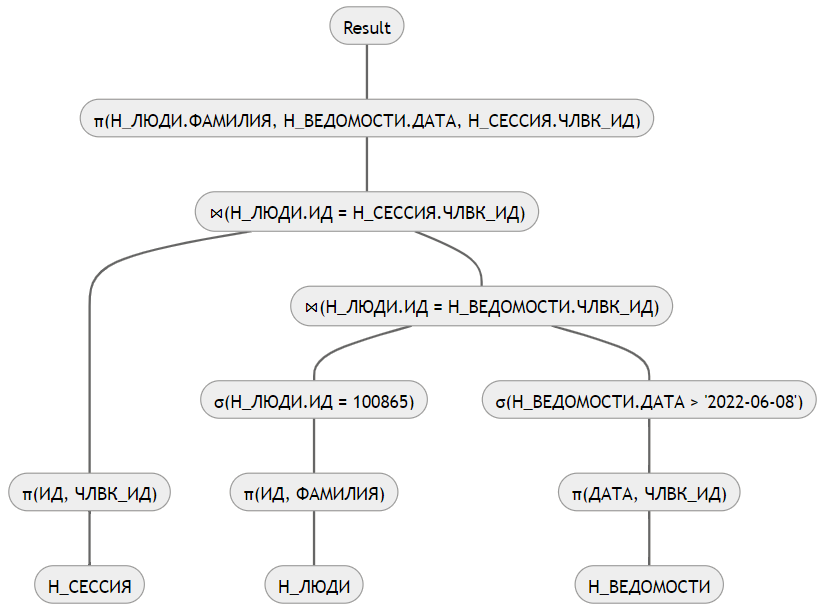
****

Оптимальным планом является план 1, так как в нем выборка и проекция производится как можно раньше, следовательно, в соединении таблиц участвует меньшее количество строк, чем в менее оптимальном плане 2.

При добавлении индексов план выполнения запроса не изменится, так как индексы только ускорят фильтрацию и соединение таблиц. В результате время выполнения запроса сократится. Но, стоит отметить, что СУБД выполняет проекцию перед выполнением фильтрации.

**Запрос 2.**

План 1.

****

План 2.



Оптимальным планом является план 1, так как в нем выборка и проекция производится как можно раньше, следовательно, в соединении таблиц участвует меньшее количество строк, чем в менее оптимальном плане 2.

При добавлении индексов план выполнения запроса не изменится, так как индексы только ускорят фильтрацию и соединение таблиц. В результате время выполнения запроса сократится. Но, стоит отметить, что СУБД выполняет проекцию перед выполнением фильтрации.

# Вывод

В рамках данной лабораторной работы были изучены индексы в PostgreSQL. Получены и отработаны на практике знания по построению планов запросов. Проведен анализ команды EXPLAIN ANALYZE.